**UNIT-1**

**1) Define DBMS?**

A) Database management system is a software which is used to manage the database.

DBMS provides an interface (interaction) to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.

For example: MySQL, Oracle, etc are commercial database which is used in different applications.

**2) What are the characteristics of Data base?**

* Real-world entity
* Relation-based tables
* Isolation of data and application
* Less redundancy
* Consistency
* Query Language

**3) What are the classification of RDBMS?**

A)The classification of RDBMS are

1.Database Administrator (DBA)

2. Naive / Parametric End Users

3. System Analyst

4. Sophisticated Users

5. Data Base Designers

6. Application Program

7. Casual Users / Temporary Users

**4) Define Metadata? (GOOGLE)**

**Ans)** Metadata is simply defined as data about data . It means it is a description and context of data . It helps to find ,organize and Understand the data

Example: When you take a photo the metadata are its File name, Size of the file, Date and time, Camera setting ,etc

**5) Defination Data Abstraction?(GOOGLE)**

**Ans)** Data Abstraction is the process of hiding the irrelevent (unwanted) data from the end user . There are three levels of Abstraction

for DBMS they are

> Physical level: This is the lowest level of data abstraction which describes how data is actually stored in database.

>**Logical** level: This middel levelof data abstraction which describes what data stored in database.

>View level: This is Highest level of data abstraction which describes the user interaction with database system.

**6)Advantages of DBMS?**

1. **Controls database redundancy:** It can control data redundancy (not so longer needed data ).
2. **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
3. **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
4. **Reduce time:** It reduces development time and maintenance need.
5. **Backup:** It provides backup and recovery subsystems ,which create automatic backup of data from hardware and software failures and restores the data if required.
6. **Multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces

**7) When not to use DBMS?(G)**

A)High untitial investement in hardware , software and haning

The Generality that a DBMS provides for defination and processing data

Overhead for providing security concurrency control recovery and integrity functions

Simple will defined database Application that are not expected to change at all

**8)List the types of End-users?(same as 15)**

**Database Administrator (DBA) :**

Database Administrator (DBA) is a person/team who defines the schema and also controls the 3 levels of database.

DBA is also responsible for providing security to the data base and he allows only the authorized users to access/modify the data base.

**Naive / Parametric End Users :**Parametric (Belong to) End Users are the unsophisticated (Unknown) who don’t have any DBMS knowledge but they frequently use the data base applications in their daily life to get the desired results.

For examples, Railway’s ticket booking users are naive users

**Casual Users / Temporary Users :**Casual Users are the users who occasionally use/access the data base but each time when they access the data base they require the new information, for example, Middle or higher level manager.

**Sophisticated Users :**Sophisticated users can be engineers, scientists, business analyst, They can develop their own data base applications according to their requirement.

**System Analyst :**

System Analyst is a user who analyzes the requirements of parametric end users. They check whether all the requirements of end users are satisfied.

**Data Base Designers :**

Data Base Designers are the users who design the structure of data base which includes tables, indexes, views, constraints, triggers, stored procedures.

**Application Program :**

Application Program are the back end programmers who writes the code for the application programs.

**9) What is data model?**

Ans)The **Data Model** is defined as an abstract model that organizes data description, data semantics, and consistency constraints of data.

**10. Describe schema and instances?**

A)The overall design of a database is called schema.

A database schema is the skeleton structure of the database. It represents the logical view of the entire database.

A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.

Example





**Instance** is the situation where a data or information is stored in the database at a particular moment of time is called an instance

11. List the categories of data models?

1. **Conceptual Data Model:**A **Conceptual Data Model** is an organized view of database concepts and their relationships.
2. **Logical Data Model:**The **Logical Data Model** is used to define the structure of data elements and to set relationships between them.
3. **Physical Data Model (Low level)**A **Physical Data Model** describes a database-specific implementation of the data model.
4. **Entity-Relationship Data Model:**
5. An ER model is the logical representation of data as objects and relationships among them.
6. **Relational Data Model:** This type of model designs the data in the form of rows and columns within a table.
7. **Object-based Data Model:** An extension of the ER model with notions of functions, encapsulation, and object identity, as well

12. What is database state?

A) A Database state is a set of stored data in a database.Changes in the database state can be caused by entering , modifying and deleting information

* for example, the STUDENT construct will contain the set of individual student entities (records) as its instances.

13. Briefly explain characteristic of database?

A)**Real-world entity** − A modern DBMS is more realistic and uses real-world entities (object , body) to design its architecture. It uses the behavior and attributes

. For example, a school database may use students as an entity and their age as an attribute.

**Relation-based tables** − DBMS allows entities and relations among them , to form tables. A user can understand the architecture of a database just by looking at the table names.

**Isolation of data and application** − A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes.

**Less redundancy** − DBMS follows the rules of normalization(Normalization is a mathematically rich and scientific process that reduces data redundancy), which splits a relation when any of its attributes is having redundancy (not so longer needed data or useless data) in

values

**Consistency** − Consistency is a state where every relation in a database remains consistent (Uniformity) . A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems .

**Query Language** − DBMS is equipped with(based on) query language, which makes it more efficient to retrieve (recive) and manipulate (modify) data. A user can apply as many and as different filtering options as required to retrieve a set of data.

14. Briefly explain Advantages of database?

A)Advantages of database are

* **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
* **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
* **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
* **Reduce time:** It reduces development time and maintenance need.
* **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
* **Multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces

15. Briefly explain about Actors on the scene? (same as 8)

A)**Database Administrator (DBA) :**

Database Administrator (DBA) is a person/team who defines the schema and also controls the 3 levels of database.

DBA is also responsible for providing security to the data base and he allows only the authorized users to access/modify the data base.

**Naive / Parametric End Users :**Parametric (Belong to) End Users are the unsophisticated (Unknown) who don’t have any DBMS knowledge but they frequently use the data base applications in their daily life to get the desired results.

For examples, Railway’s ticket booking users are naive users

**Casual Users / Temporary Users :**Casual Users are the users who occasionally use/access the data base but each time when they access the data base they require the new information, for example, Middle or higher level manager.

**Sophisticated Users :**Sophisticated users can be engineers, scientists, business analyst, They can develop their own data base applications according to their requirement.

**System Analyst :**

System Analyst is a user who analyzes the requirements of parametric end users. They check whether all the requirements of end users are satisfied.

**Data Base Designers :**

Data Base Designers are the users who design the structure of data base which includes tables, indexes, views, constraints, triggers, stored procedures.

**Application Program :**

Application Program are the back end programmers who writes the code for the application programs.

16. With neat diagram explain 3 Schema Architecture?

A)



**1.External or View level:**

* This is the highest level of database abstraction.
* External or view level describes the actual view of data that is relevant to the particular user.

**2.Conceptual or Logical level:**

* The conceptual level describes the structure of the whole database.
* This level acts as a middle layer between the physical storage and user view.

**3.Internal or Physical level:**

* This is the lowest level of database abstraction.
* It describes how the data is actually stored in the database and provides methods to access data from the database.

**17.What is data independence and explain their types?**

A)Data independence refers characteristic of being able to modify the schema at one level of the database system without altering the schema at the next higher level.

**1.Logical Data Independence**

Logical data independence refers characteristic of being able to change the conceptual schema without having to change the external schema.

**2. Physical Data Independence**

Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.

18. Briefly explain DBMS languages?

A)There are 4 types of DBMS languages

**1.Data Definition Language:**

**DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.

Some tasks that come under DDL are Create, Alter, Drop , Rename , Comment ,etc.

**2. Data Manipulation Language:DML** stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

**3.Data Control Language:DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.

**4. Transaction Control Language:**TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

19. List and explain DBMS interfaces?

A)**1. Menu-Based Interfaces for Web Clients or Browsing –**  
These interfaces present(gives) withlists of options (called menus) that lead the user through the formation of a request.

**2. Forms-Based Interfaces –**

A forms-based interface displays a form to each user. A user either can fill the required information in the form or create a new form to get the data .

**3. Graphical User Interface –**

A GUI typically displays a schema to the user in diagrammatic form.

The user then can specify a query by manipulating the diagram

**4. Natural language Interfaces –**

A Natural language interface has its own schema, which is similar to the database conceptual schema as well as a dictionary of important words.

**5. Interfaces for DBA –**

These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, reorganizing the storage structures of a databases.

**6. Interfaces for Parametric Users.**

Parametric users, such as bank tellers, often have a small set of operations that they must perform repeatedly.

20. Explain Centralized Architectures of DBMS

A)Architectures for DBMSs have followed trends similar to those for general computer system architectures.

* Earlier architectures used mainframe computers to provide the main processing for all system functions, including user application programs and user interface programs, as well as all the DBMS functionality.
* As prices of hardware declined (decrease) , most users replaced their terminals with PCs and workstations.
* At first, database systems used these computers similarly to how they had used display terminals, so that the DBMS itself was still a **centralized** DBMS in which all the DBMS functionality, application program execution, and user inter-face processing were carried out on one machine. DBMS systems started to exploit ( ) the available processing power at the user side,

21. Explain basic Client-Server architecture of DBMS.

A)**Client-server architecture**, architecture of a computer network in which many clients (remote processors) request and receive service from a centralized server (host computer).

Client computers provide an interface to allow a computer user to request services of the server and to display the results the server returns. Servers wait for requests to arrive from clients and then respond to them

Types of Client Server Architecture

1 Tier Architecture

2 Tier Architecture

3 Tier Architecture

22. Explain Two-Tier architecture of DBMS

A)



A **2 Tier Architecture** in DBMS is a Database architecture where the presentation layer runs on a client (PC, Mobile, Tablet, etc.), and data is stored on a server called the second tier.

Two tier architecture provides added security to the DBMS as it is not exposed to the end-user directly.

It also provides direct and faster communication.

23. Explain Three-Tier architecture of DBMS.

A)



A **3 Tier Architecture** in DBMS is the most popular client server architecture in DBMS in which the development and maintenance of functional processes, logic, data access, data storage, and user interface is done independently as separate modules.

* Three Tier architecture contains a presentation layer, an application layer, and a database server.

24. What are the classifications of database management system?

* A)The first is the **data model** on which the DBMS is based.
* The main data model used in many current commercial DBMSs is the **relational data model**.
* The **object data model** has been implemented in some commercial systems but has not had widespread use.

Many legacy applications still run on database systems based on the **hierarchical** and **network data** **models**.

* Single-user systems support only one user at a time and are mostly used with PCsMultiuser systems, which include the majority of DBMSs, support con-current multiple users.

25. Briefly Describe database System Utilities?

**A)**

System Utilities provide basic functioning to users so that they do not need to write their own environment for program development (editors, compilers) and program execution (shells).

In some sense, they are bundles of useful system calls.

**Examples:** operating system, interpreter, compiler, editors …..Etc are all system programs.

They can be divided into these categories:

1. **File Management:** These programs create, delete, copy, rename, print and generally manipulate files and directories.
2. **Status information:** some programs simply ask the system for the date, time, and amount of available memory or disk space, number of users.
3. **File Modification:** several text editors may be available to create and modify the content of files stored on disk or other storage devices.
4. **Programming language support:** compilers, assemblers debuggers and interpreters
5. **Program loading and execution:** once a program is assembled or compiled, it must be loaded into the memory to be executed.
6. **Communications:** with help of these programs we can send message to one pc to other pc by remote log in.

**UNIT 3 Question Bank**

1. **Define the term Domain, Attribute, Tuple & Relation.**

**Domain:**

* + The term domain refers to the set of values found under an attribute name.
  + It is the value under a column.

**Attribute:** It is a column of a table.

**Tuple:** It is a row. One row in a table is known as a tuple.

**Relation:**A relation is a two-dimensional table i.e the information is arranged in rows and columns.

1. **What are the main categories of constraints?**

Constraints on databases is divided into 3 main categories,

1. Inherent model-based constraints / Implicit Constraint
2. Schema-based constraints / Explicit constraints
3. Application based / Semantic constraints / Business Rules
4. **List different keys in RDBMS?**

* Candidate Key
* Super Key
* Primary Key
* Alternate Key
* Foreign Key

1. **What is Domain Constraint?**

* Domain constraints specify that within each tuple, the value of each attribute A must be an atomic value from the domain dom(A).

**Examples for domains:** Phone\_Numbers: The set of 10 digit phone numbers is valid in India.

1. **Write syntax and example for CREATE table statement.**

**Description**

The CREATE TABLE statement allows you to create and define a table.

**Syntax**

CREATE TABLE table\_name (

column1 datatype [ NULL | NOT NULL ], column2 datatype [ NULL | NOT NULL ],

...

column\_n datatype [ NULL | NOT NULL ]

);

**Example**

CREATE TABLE STUDENT (

studentid int NOT NULL, studentname varchar(50) NOT NULL, city varchar(50),

mobno int

);

1. **Write syntax and example for INSERT table statement.**

**Description**

The INSERT statement is used to insert a single record or multiple records into a table in Oracle.

INSERT INTO table VALUES

(expression1, expression2, ... expression\_n );

**Example**

INSERT INTO STUDENT

(studentid,studentname,city,mobno) VALUES

(101,’James willium’,’San-Fransisco’,4589631112);

1. **Write syntax and example for UPDATE table statement.**

The UPDATE statement is used to update existing records in a table in an Oracle database.

UPDATE table

SET column1 = expression1, column2 = expression2,

...

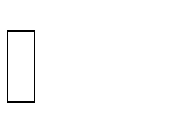
column\_n = expression\_n [WHERE conditions];

**Example**

UPDATE BCASTUDENT

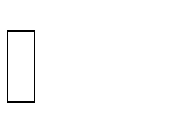
SET studentname = 'Anderson' WHERE studentid = 1;

1. **List Unary Relational Operations**
2. SELECT (symbol: ϭ(sigma))
3. PROJECT (symbol: π (pi))
4. RENAME (symbol:  (rho))
5. **List Relational Algebra Operations from Set Theory**
6. UNION (U)
7. INTERSECTION
8. DIFFERENCE (MINUs or -)
9. CARTESIAN PRODUCT (X)
10. **Write a syntax and example for SELECT and PROJECT.**

The SELECT operation is used to choose a subset of the tuples from a relation that satisfies the select condition.



**ϭ** SALARY > 30,000 (EMPLOYEE)

The PROJECT operation, on the other hand selects certain columns from the table and discarding other columns.

π<attribute list>(R)

π LNAME, FNAME,SALARY(EMPLOYEE)

1. **Define Assertion.**

An **assertion** is a statement in **SQL** that ensures a certain condition will always exist in the database. **Assertions** are like column and table constraints, except that they are specified separately from table definitions.

1. **Define Triggers.**

A **trigger** is a stored procedure in **database** which automatically invokes whenever a special event in the **database** occurs.

For **example**, a **trigger** can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

1. **Define View in SQL**.

A view is a “virtual” table that is derived from other tables.

* SQL command: **CREATE VIEW**
  + a table (view) name
  + a possible list of attribute names.
  + A query to specify the table contents.

1. **Define fragmentation.**

Fragmentation is the task of dividing a table into a set of smaller tables. The subsets of the table are called **fragments**. Fragmentation can be of three types: horizontal, vertical, and hybrid (combination of horizontal and vertical).

1. **List various attributes of Indexing.**

The indexing has various attributes:

* **Access Types**
* **Access Time**
* **Insertion Time**
* **Deletion Time**
* **Space Overhead**

**Marks**

1. **Explain the characteristics of relations?**
   * **Ordering of tuples in a relation**

A relation is defined as set of tuples.

* + - Mathematically, **elements of a set have no order among them;**

hence tuples in a relation do not have any particular order.

* + **Ordering of values within a tuple**
    - According to the previous definition of a relation, an n-tuple is an ordered list of n values, **the ordering of values in a tuple and attributes in a relation schema is important.**
  + **Values and NULLs in the tuples**
    - **Each value in a tuple is atomic value. [composite and multivalued attributes are not allowed].**
    - This model is sometimes called as **flat relational model.**
    - **The NULL values are used to represent the values of attributes thatmay be unknown or may not apply to a tuple.**
    - Several meaning for NULL values: Value unknown, value exists but not available, attribute doesn’t apply.
  + **Interpretation of a relation**
    - The relation schema can be interpreted as declaration or a type of assertion.
    - **For example, the schema of the STUDENT relation asserts that, a student entity has a Name, Roll#, Mobile.. Etc attributes.**
    - Each tuple in the relation can then be interpreted as a **fact** or a particular instance of the assertion.
    - For example, the first tuple in the STUDENT relation asserts the

fact that there is a student whose name is Prakash, Roll# 10 and etc.

1. **Explain schema based constraints.**

Constraints that can be directly expressed in schemas of the data model, typically by specifying them in DDL, we call this as schema based constraints.

* + **The schema based constraints include,**

1. Domain Constraints
2. Key Constraints
3. Constraints on NULLs
4. Entity Integrity Constraints
5. Referential Integrity Constraints
6. **Domain Constraints**
   * **Domain constraint defines the domain or set of values for an attribute.**
   * It specifies that the value taken by the attribute must be the atomic value from its domain.
   * Ex: Phone\_Numbers attribute take 10 digit.
7. **Key Constraints**
   * **An attribute that can uniquely identify a tuple in a relation is called the key of the table.** The value of the attribute for different tuples in

the relation has to be unique.

* + Ex: CustomerID is a key attribute of Customer Table.

1. **Constraints on NULLs**
   * **If we want to have all the tuples should have values, we use NOT**

NULL constraint.

* + Ex: Sutudent name not null constraint

1. **Entity Integrity Constraints**

The entity integrity constraint states **that no primary key value can be NULL**

1. **Referential integrity constraints**

The **referential integrity** constraint is specified between two relations and is used to maintain the consistency among the tuples in two relations.

1. **Explain the different types of keys with example.**
   * **Candidate Key**

**Candidate key is a key of a table which can be selected as a primary key of the table.** A table can have multiple candidate keys, out of which one can be selected as a primary key.

**Example**: Employee\_Id, License\_Number and Passport\_Number are candidate keys

* + **Primary Key**

**Primary key does not allow null value in the column and keeps unique values throughout the column.** Example,

**Example:** Employee\_Id is a primary key of Employee table.

* + **Unique Key**

**Unique key is similar to primary key and does not allow duplicate values in the column**. But Unique key allows one null value in the column.

* + **Alternate Key**

**Alternate key is a candidate key, currently not selected as primary key of the table**.

**Example:** License\_Number and Passport\_Number are alternate keys.

* + **Composite Key**

**Composite key (also known as compound key or concatenated key) is a group of two or more columns that identifies each row of a table uniquely**. It can be a primary key or candidate key also.

**Example:** In salary table, Employee\_Id and Salary\_Month\_Year are combined together to identify each row uniquely in Salary table. Independently Employee\_Id or Salary\_Month\_Year column cannot identify each row uniquely.

* + **Super Key**

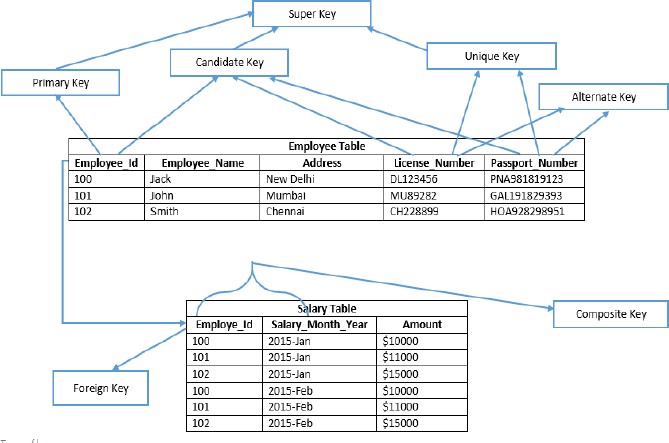
**Super key is a set of columns on which all columns of the table are functionally dependent.** It is a set of columns that uniquely identifies each row in a table. For **example:** {Employee\_Id}, {Employee\_Id, Employee\_Name},

{Employee\_Id, Employee\_Name, Address} etc.

* + **Foreign Key**

In a relationship between two tables, **a primary key of one table is referred as a foreign key in another table**. Foreign key can have duplicate values in it and can also keep null values if column is defined to accept nulls.

**Example:** Employee\_Id (primary key of Employee table ) is a foreign key in Salary table.



**1.Explain the concept of JOINS in detail.**

* 1. **Joins can be simply defined as the combining or merging the related tuples from the two different relations into a single type.**
  2. The sequence of CARTESIAN PRODECT followed by SELECT is used quite commonly to identify and select related tuples from two relations
  3. **The general form of a join operation on two relations R(A1, A2, . . ., An) and S(B1, B2, . . ., Bm) is:**

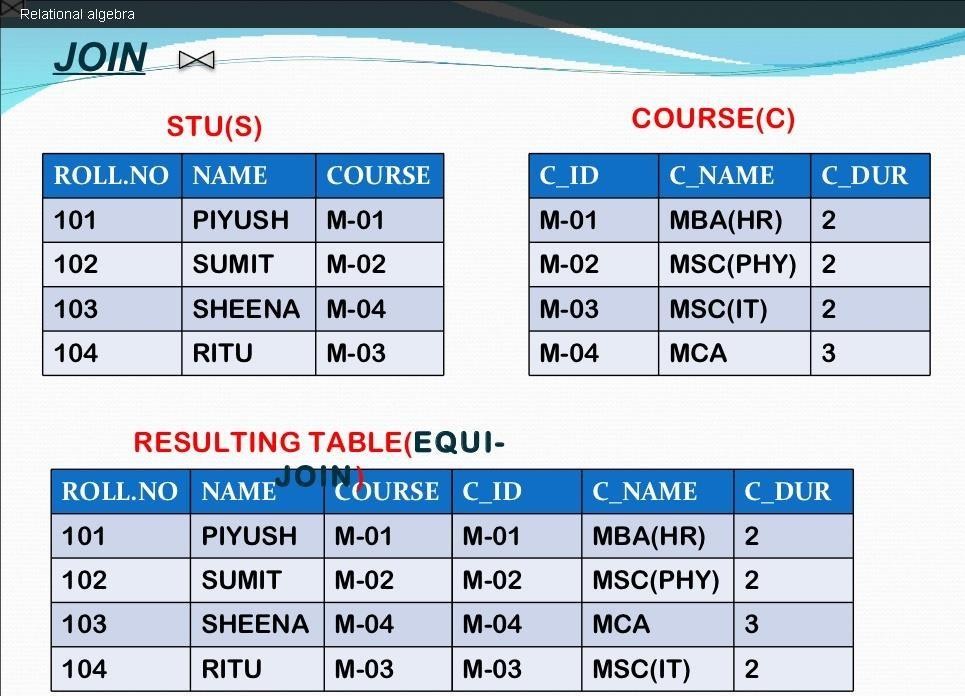


**where R and S can be any relations that result from general**

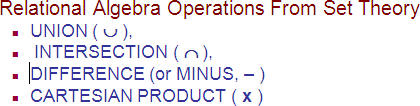
***relational algebra expressions*.**

**We can classify joins basically into two types:**

* + 1. **INNER JOINS**: These joins are the one that has the tuples that satisfy some conditions and rest are discarded. Further they are classified as
* Theta join
* Equi join
* Natural join
  + 1. **OUTER JOINS**: These have all the tuples from either or both the relations. Further they are classified as
* Left outer join
* Right outer join
* Full outer join



**2.Explain the concept of set theory in relational algebra?**



**UNION:**

* **Binary operation, denoted by U**
* **The result of RUS is a relation that includes all the tuples that are either in R or in S or in both R and S**
* **Duplicate tuples are eliminated.**

**INTERSECTION:**

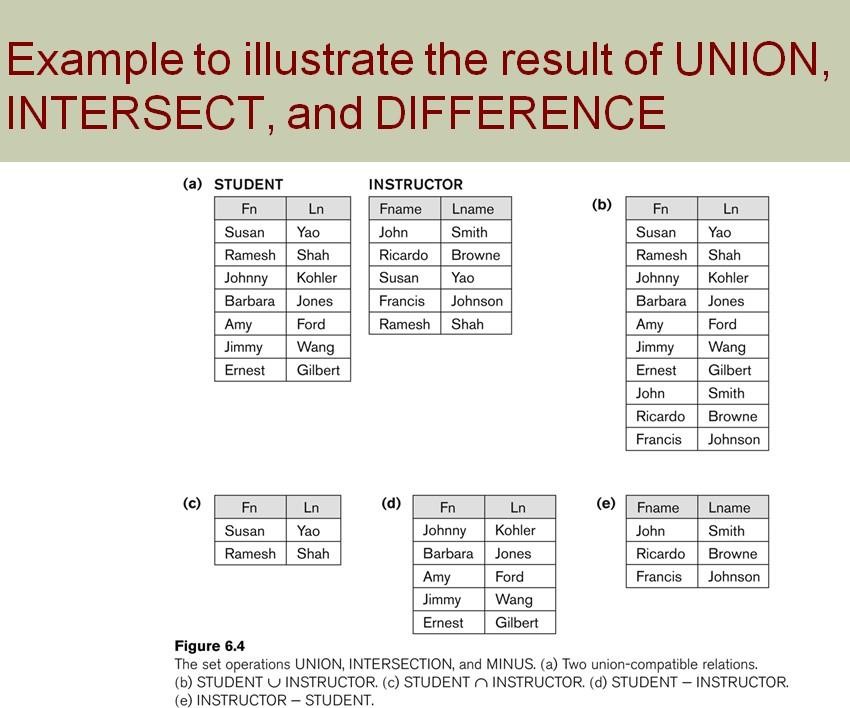
* **Intersection is denoted by**
* **The result of the operation R S, is a relation that includes all tuples that are in both R and S**

**SET DIFFERENCE:**

* **Set difference, denoted by Minus(-)**
* **The result of R-S, is a relation that includes all tuples that are in R but not in S.**
* **The attributes names in the result will be the same as the attribute names in R.**

**CARTESIAN PRODUCT:**

* **Cartesian product, denoted by (X)**
* **This operation is used to combine tuples from two relations in a combinatorial fashion.**
* **Denoted by R(A1,A2,…..,An) x S(B1,B2,…,Bm)**



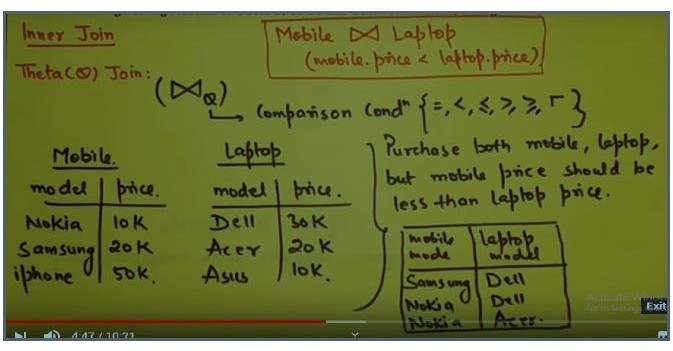
1. **Explain the concept of INNER JOINS in detail.**

**INNER JOINS**: These joins are the one that has the tuples that satisfy some conditions and rest are discarded . Further they are classified as

* + Theta join
  + Equi join
  + Natural join

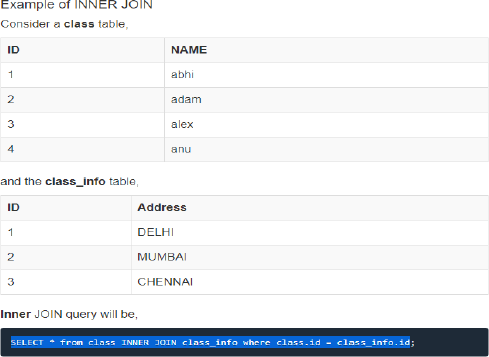
**Theta join**

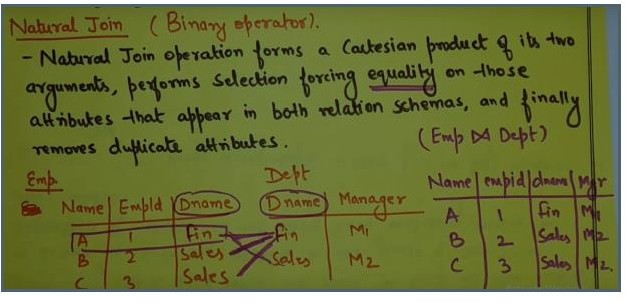
The general case of JOIN operation is called a Theta-join



**Equi join**

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the SQL query.





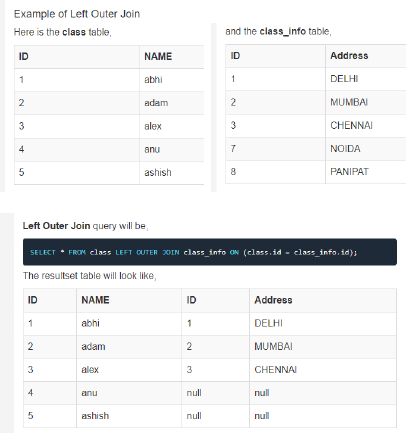
1. **Explain Outer Join.**

**OUTER JOIN**

Outer Join is based on both matched and unmatched data. Outer Joins subdivide further into,

1. **Left Outer Join**
2. **Right Outer Join**
3. **Full Outer Join**

**LEFT Outer Join**

The left outer join returns a result set table with the **matched data** from the two tables and then the remaining rows of the **left** table and null from the **right** table's columns.

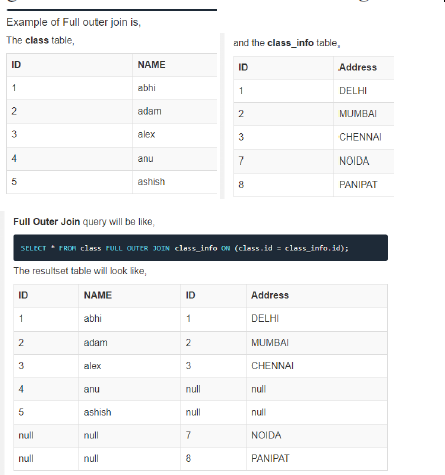
**RIGHT Outer Join**

The right outer join returns a resultset table with the **matched data** from the two tables being joined, then the remaining rows of the **right** table and null for the remaining **left** table's columns.



**Full Outer Join**

The full outer join returns a result set table with the **matched data** of two table then remaining rows of both **left** table and then the **right** table.



1. **Explain Group Functions.**

Group functions are built-in SQL functions that operate on groups of rows and return one value for the entire group. These functions are: **COUNT, MAX, MIN, AVG, SUM, DISTINCT**

* **SQL COUNT ():** This function returns the number of rows in the table that satisfies the condition specified in the WHERE condition. If the WHERE condition is not specified, then the query returns the total number of rows in the table.

**For Example:** If you want the number of employees in a particular department, the query would be:

SELECT COUNT(\*) from employee where department=’Electronic’;

* **SQL DISTINCT():** This function is used to select the distinct rows.

**For Example:**

SELECT DISTINCT dept FROM employee;

* **SQL MAX():** This function is used to get the maximum value from a column.

**For Example:** SELECT MAX (salary) FROM employee;

* **SQL MIN():** This function is used to get the minimum value from a column.

**For Example:** SELECT MIN (salary) FROM employee;

* **SQL AVG():** This function is used to get the average value of a numeric column.

**For Example:** SELECT AVG (salary) FROM employee;

* **SQL SUM():** This function is used to get the sum of a numeric column

**For Example:** SELECT SUM (salary) FROM employee;

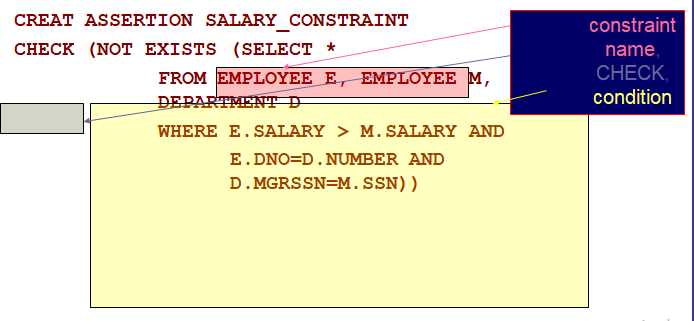
1. **Explain Assertion constraint in SQL with example.**

An **assertion** is a statement in **SQL** that ensures a certain condition will always exist in the database. **Assertions** are like column and table constraints, except that they are specified separately from table definitions.

* + Components include:
    - a constraint name,
    - followed by CHECK,
    - followed by a condition

**An Example**

* + “The salary of an employee must not be greater than the salary of the manager of the department that the employee works for’’



1. **Explain Triggers with example.**

A **trigger** is a stored procedure in **database** which automatically invokes whenever a special event in the **database** occurs. For **example**, a **trigger** can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

**Objective:** to monitor a database and take initiate action when a condition occurs.

**Syntax:**

create trigger [trigger\_name] [before | after]

{insert | update | delete} on [table\_name]

[for each row]

[trigger\_body]

**Example:**

**create table student( studentid int primary key, marks int**

**);**

**create trigger calculate before insert**

**on student for each row begin**

**:new.marks:= :new.marks + 100; end;**

**/ OR**

**create or replace trigger stud\_trigger before insert on student**

**for each row declare**

**no number(5); begin**

**select max(stude\_id)into no from student; no:=no+1;**

**:new.stude\_id:=no; end;**

**/**

1. **Explain Views with example.**

A view is a “virtual” table that is derived from other tables allows for limited update operations

1. Since the table may not physically be stored.

SQL command: **CREATE VIEW**

1. a table (view) name
2. a possible list of attribute names (for example, when arithmetic operations are specified or when we want the names to be different from the attributes in the base relations)
3. a query to specify the table contents

**Syntax:**

CREATE VIEW view\_name AS SELECT column1, column2.....

FROM table\_name WHERE condition;

**An Example:**

* **Specify a different WORKS\_ON table** CREATE VIEW WORKS\_ON\_NEW AS SELECT FNAME, LNAME, PNAME, HOURS FROM EMPLOYEE, PROJECT, WORKS\_ON WHERE SSN=ESSN AND PNO=PNUMBER GROUP BY PNAME;

1. **Explain Fragmentation with its types.**

Fragmentation is the task of dividing a table into a set of smaller tables.

The subsets of the table are called **fragments.**

**Fragmentation can be of three types:**

1. **Horizontal,**
2. **Vertical**
3. **Hybrid (combination of horizontal and vertical).**
   * **Horizontal fragmentation** groups the tuples of a table in accordance to values of one or more fields.

**For example**, in the student schema, if the details of all students of Computer Science Course needs to be maintained at the School of Computer Science, then the designer will horizontally fragment the database as follows −

**CREATE COMP\_STD AS SELECT \* FROM STUDENT WHERE COURSE =**

**"Computer Science";**

* + In **vertical fragmentation,** the fields or columns of a table are grouped into fragments. In order to maintain re-constructiveness, each fragment should contain the primary key field(s) of the table.

Pink tissue paper**For example**, let us consider that a University database keeps records of all registered students in a Student table having the following schema.

Now, the fees details are maintained in the accounts section. In this case, the designer will fragment the database as follows −

**CREATE TABLE STD\_FEES AS SELECT Regd\_No, Fees FROM STUDENT;**

* In **hybrid fragmentation**, a combination of horizontal and vertical fragmentation techniques are used. This is the most flexible fragmentation technique since it generates fragments with minimal extraneous information. However, reconstruction of the original table is often an expensive task.
  + **Hybrid fragmentation can be done in two alternative ways −**
    - At first, generate a set of horizontal fragments; then generate

vertical fragments from one or more of the horizontal fragments.

* + - At first, generate a set of vertical fragments; then generate horizontal fragments from one or more of the vertical fragments

1. **Explain Hash file Organization.**

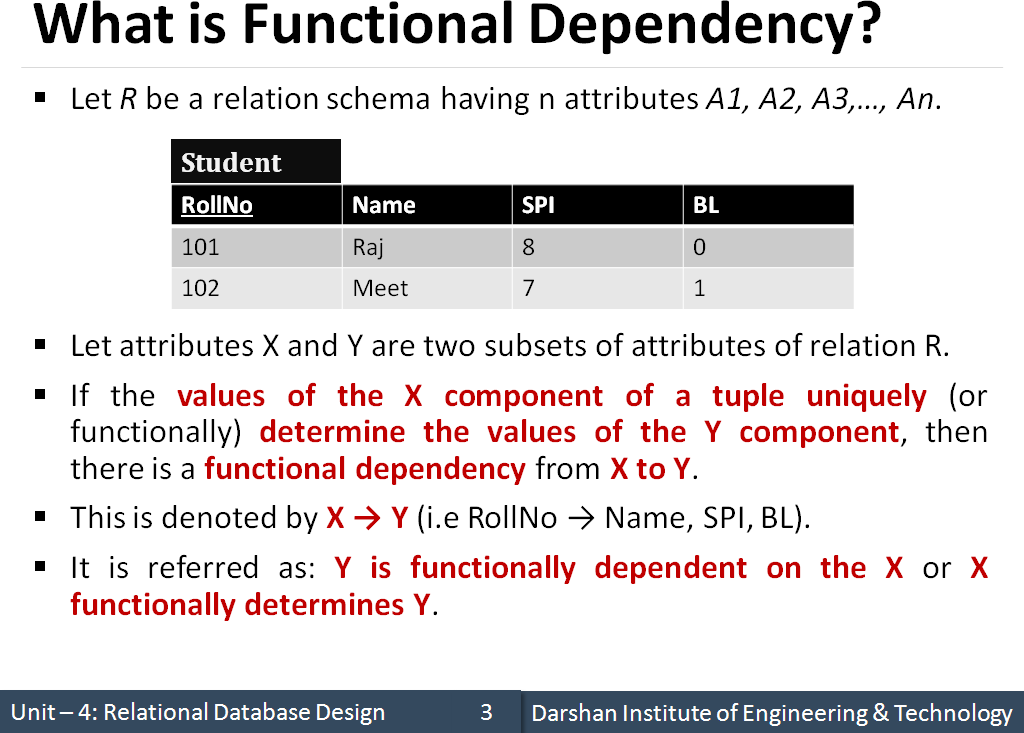
Indices are based on the values being distributed uniformly across a range of buckets. The bucket to which a value is assigned is determined by a function called a hash function.

**There are primarily three methods of indexing:**

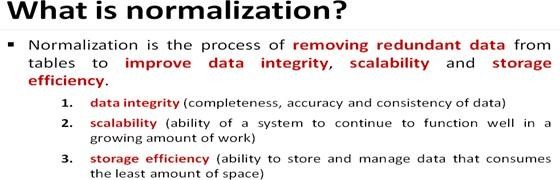
* Clustered Indexing
* Non-Clustered or Secondary Indexing
* Multilevel Indexing
* **Clustered Indexing**
  + When more than two records are stored in the same file these types of storing known as **cluster indexing.**
  + By using the cluster indexing we can reduce the cost of searching reason being multiple records related to the same thing are stored at one place and it also gives the frequent join of more than two tables(records).
* **Non-Clustered or Secondary Indexing**
  + A non clustered index just tells us where the data lies, i.e. it gives us a list of virtual pointers or references to the location where the data is actually stored. Data is not physically stored in the order of the index. Instead, data is present in leaf nodes
* **Multilevel Indexing**
  + With the growth of the size of the database, indices also grow. As the index is stored in the main memory, a single-level index might become too large a size to store with multiple disk accesses. The multilevel indexing segregates the main block into various smaller blocks so that the same can stored in a single block. The outer blocks are divided into inner blocks which in turn are pointed to the data blocks. This can be easily stored in the main memory with fewer overheads.

**UNIT 4**

1. **marks**
   1. **What is Functional Dependency?**



* 1. **What is normalization?**



* 1. **List the types of functional dependency.**
     + Full Functional dependency
     + Partial Functional dependency
     + Transitive Functional dependency
     + Trivial Functional dependency
     + Non-Trivial Functional dependency
  2. **What is Transitive dependency?**
* A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies.
* X -> Z is a transitive dependency if the following three functional dependencies hold true:

X->Y

Y does not ->X Y->Z

* **Example**:

Book} ->{Author} (if we know the book, we knows the author name)

{Author} does not ->{Book}

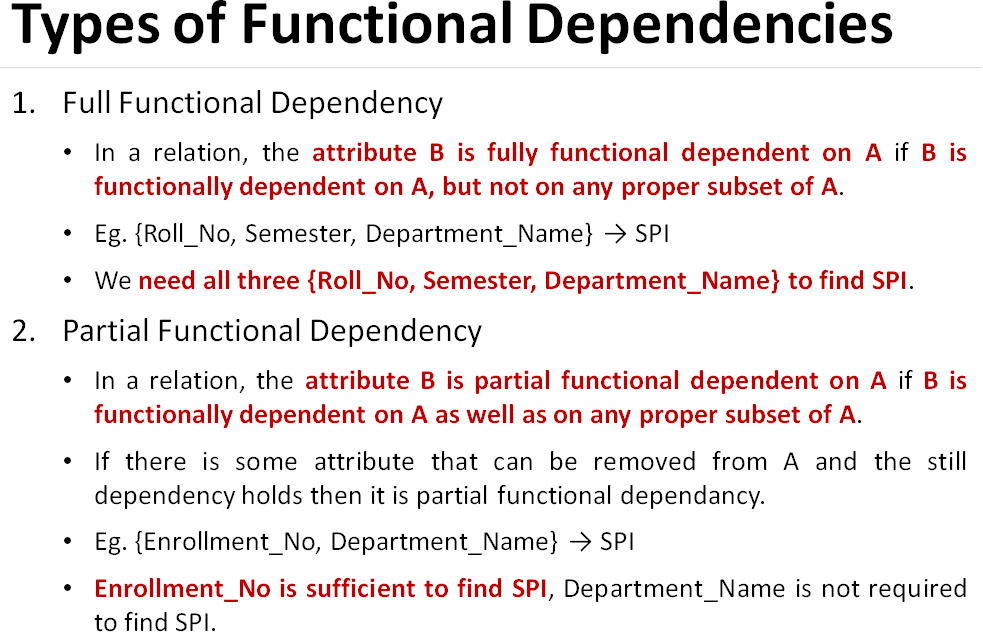
{Author} -> {Author\_age}

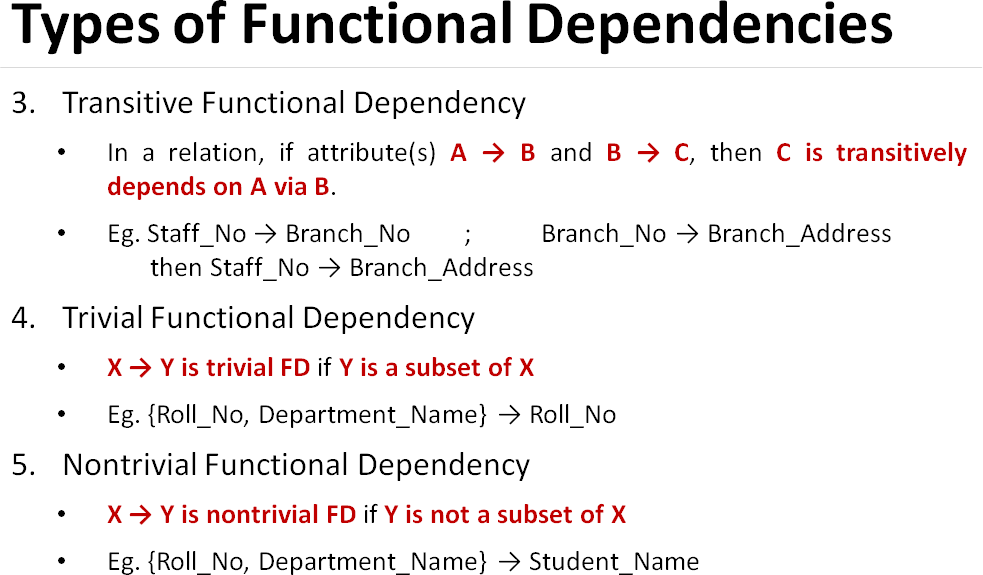
* 1. **What is anomaly in database design?**
* Anomalies are problems that can occur in poorly planned, unnormalized database where all the data are stored in one table.
* There are three types of anomalies that can arise in database because of redundancy:
  + **Insert anomaly**
  + **Delete anomaly**
  + **Update / Modification anomaly**
  1. **List types of anomalies that arise in DBMS?**
* **Insert anomaly**
* **Delete anomaly**
* **Update / Modification anomaly**

**5 Marks**

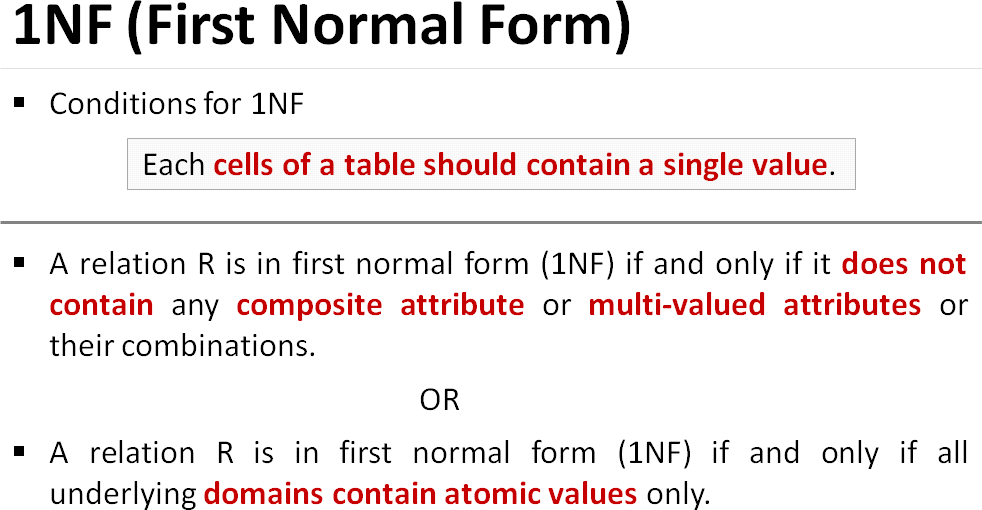
1. **Explain Functional Dependency.**

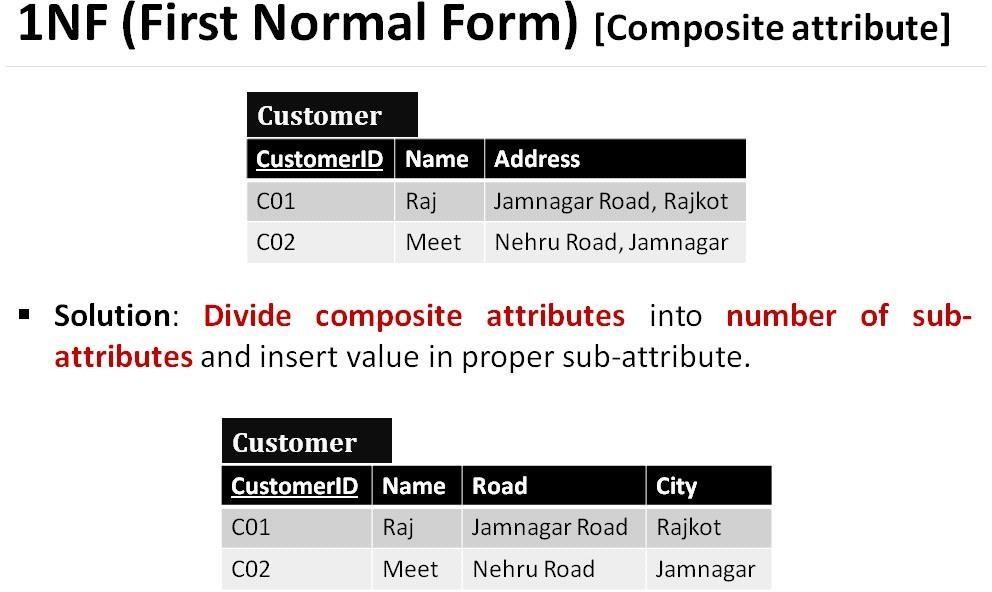
**Functional dependency** is a relationship that exists when one attribute uniquely determines another attribute. If R is a relation with attributes X and Y, a **functional dependency** between the attributes is represented as X->Y, which specifies Y is **functionally** dependent on X.

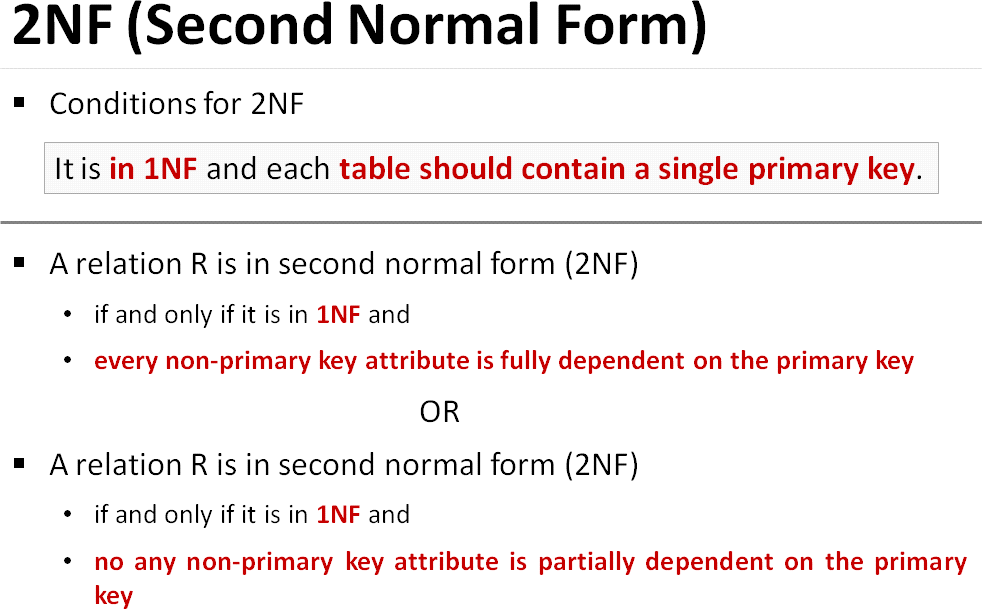


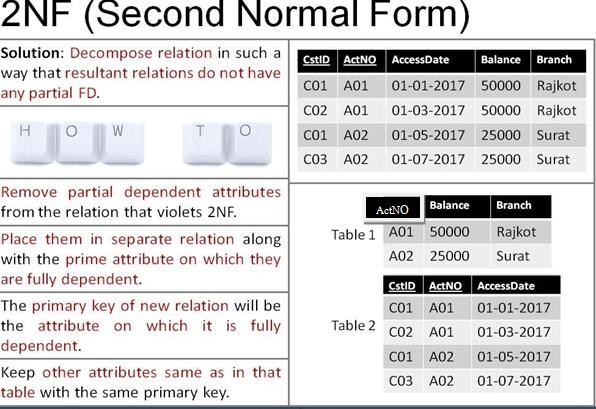


1. **Explain 1 NF and 2 NF with example**









1. **Explain 3NF and BCNF with example.**

